Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Applied Logic 2 (2004) 219–239

JOURNAL OF APPLIED LOGIC

www.elsevier.com/locate/jal

An experiment concerning mathematical proofs on computers with French undergraduate students

R. David, C. Raffalli *

Université de Savoie, Laboratoire de Mathématiques, Le Bourget du Lac Cedex 73376, France

Available online 17 April 2004

1. Introduction

The undergraduate students in mathematics of the University of Chambéry have the opportunity of an optional course in logic during their third year. Some years ago, this course was a classical introduction to logic as, for example, in [3]. Due to the difficulties of the students, it became, step by step, a training course in mathematical reasoning. The use of PhoX, the proof assistant developed by C. Raffalli (see [14]), allowed to carry out, in detail, "real" proofs during lab sessions on computers.

We started to use PhoX with students in 1999. At the beginning, the examples we considered were very simple but, step by step, they became more and more intricate and, in 2001, the students spent about 10 hours with PhoX. During the 2002–2003 academic year, this course (i.e., 50 hours) has totally been devoted to work on computers. The students had to do, first with the computer and then on paper, the proof of three classical mathematical theorems (see Section 3.3). During the 2003–2004 academic year, only a short course (10 hours) of training to mathematical reasoning was maintained: our colleagues (mathematicians but not logicians as we are) did not consider that this course was important and they decided to suppress it. During the 2001–2003 academic years, some first year students also did some (more elementary) proofs with PhoX.

It is, as far as we know, the first time in France that math students, at this level, have done proofs of that difficulty with a computer. The goal of this paper is to describe these various experiments. We will mainly concentrate on the 2002–2003 academic year since it is, in some sense, its apogee. We also give the main important points concerning the other years and the experiment for the first year students.

This paper is not, strictly speaking, a research paper. Even though it gives some details on the way PhoX works and thus can be considered as a short introduction to this software, we are here more concerned with pedagogical matters. However, the problems we mention

* Corresponding author.
E-mail addresses: david@univ-savoie.fr (R. David), raffalli@univ-savoie.fr (C. Raffalli).

1570-8683/$ – see front matter
doi:10.1016/j.jal.2004.02.002

may interest computer science researchers working in the field of formal methods because they may help them in prototyping proof assistants designed for teaching mathematics.

2. Proof and computers

Proof assistants like ACL2 [10], COQ [8], HOL [7,9], Isabelle [12], LEGO [13], PVS [11] ... allow to do mathematical proofs with a computer and their correctness is thus guaranteed. This is not automated deduction: the user can (and generally must) guide the machine in the reasoning. The software just checks that each step of the proof is correct.

Most of the software above provides more or less sophisticated algorithms of automatic deduction allowing to finish the proof without the participation of the user. But this aspect does not have much teaching interest because the interesting point is the correctness of each step of a proof. These systems were developed by specialists primarily aiming for applications in formal verification for programs, circuits, communications protocols, etc. Learning to use these systems is not simple and clearly requires too much time for undergraduate students in mathematics.

PhoX can be used by undergraduate students because it uses the usual mathematical language (see some examples below) and the small number of commands (ten are mainly sufficient) makes its learning easy. As far as the authors know, it is the only proof assistant used in France for teaching mathematics (and not just logic).

Various other systems[1] are used to teach logic using computers. But, even though we are logicians, our goal is now to teach mathematical reasoning using a computer and avoid logic. Since, in France, most of the math students will never have a course in logic, it appears to be necessary! We give some answers to the question "why PhoX?" in Appendix D.

3. The main experiment

We describe in this section what we have done during the 2002–2003 academic year.

3.1. A bit of logic

Before starting to work with computers, we spent three or four hours on the blackboard to describe the grammar of proofs and to do some examples.

We did not give a formal presentation of the notion of formula that can be found, for example, in [3]: it is sufficiently intuitive to the students and this avoids the distinction between first and higher-order formulas. This distinction is important in a classical course of logic but such a course is not our goal here. As it is traditional in logic, we nevertheless introduced the symbols $\vee$ for "or", $\wedge$ for "and" and $\neg$ for "not". We essentially said the following:

[1] For example see http://www.cs.otago.ac.nz/staffpriv/hans/logiccourseware.html.

A proof is a sequence of *steps*: each of them corresponds to the application of a *rule*. At each step of a proof, there is a *knowledge list*[2] (denoted as KL). At the beginning of a proof, KL is empty: more precisely it contains only the results given during the course, i.e., theorems, lemmas, axioms, etc. Some rules allow to extend KL for the rest of the proof. If a proof has to consider several cases, the KL of each case are identical at the beginning but they can change independently.

A first rule simply says that if a formula belongs to the KL, then it is proved. For each logical symbol (connective and quantifier) there are two rules concerning the formulas whose main connective is the corresponding one. One, called the *introduction*, which is used to *prove* such a formula. For example, to prove $A \to B$ one adds A to the KL and one proves B. The other one, called the *elimination*, which allows to *use* such a hypothesis (or a lemma, an axiom, etc.). For example, if $A \vee B$ belongs to the KL and one wants to prove C, it is enough to prove both $A \to C$ and[3] $B \to C$. We also mentioned the absurdity rule which is, usually, not clear for them.

These rules[4] are given (using PhoX's terminology) in Table 1. It could be surprising that, with a few extra axioms like AC to specify the underlying theory, this is enough to prove any mathematical result. This is not very difficult to explain, but many mathematicians are not aware of Gödel's completeness theorem!

3.2. The introduction to the software

After these few hours, the rest of the course took place in the computer lab. The students worked in groups of two. The computer system was Linux but those who wanted to use the software at home could use a version for Windows. We gave the students a table with the main commands of PhoX (Table 1).

The first lab session took about four hours. The students proved some elementary facts in propositional calculus and in first order logic: for example $[(A \wedge B) \to C] \leftrightarrow [A \to (B \to C)]$ or $\exists x (A \wedge B) \to (\exists x A \wedge \exists x B)$.

The second lab session also took about four hours. The students showed that if two one-to-one functions (from E into E) have disjoint supports,[5] then they commute. This example is simple because it was seen during the course of algebra (when it is shown that, in $\mathcal{S}_n$, two disjoint cycles commute). After that, the students were familiar enough with PhoX and they had no major difficulties with it.

[2] This list is commonly called the *hypotheses* (or context) but this term, that logicians use most often in this sense, can be misleading because it can be thought as constant. The fact that this list is constantly changing is an important point which was not clear to the students.

[3] Note that this "and" is very surprising for the students.

[4] In the rest of the paper, the rules are given their usual name. For example, $\wedge_i$ is the introduction rule for $\wedge$.

[5] The *support* of a function from E to E is the set of points that are not mapped to themselves.

Table 1
Table of commands

Rules on	Current goal		Command	PhoX answers		In English we will say
The conclusion		$\vdash A \to B$	intro.	$H := A$	$\vdash B$	Assume A and show B.
		$\vdash A \wedge B$	intro.	(1)	$\vdash A$	Let us first show A then B
				(2)	$\vdash B$	
		$\vdash A \vee B$	intro l. (*l for left*)		$\vdash A$	It is enough to show A.
		$\vdash A \vee B$	intro r. (*r for right*)		$\vdash B$	It is enough to show B.
		$\vdash \forall x\, A[x]$	intro.		$\vdash A[x]$	Let x be any object. Show $A[x]$.
		$\vdash \exists x\, A[x]$	intro.		$\vdash A[?]$	We are looking for an x such that $A[x]$. t is the
		instance? t			$\vdash A[t]$	object we are looking for, let us show $A[t]$.
		$\vdash \neg A$	intro.	$H := A$	$\vdash$ *False*	Assume A and look for a contradiction.
A hypothesis	$H := A$	$\vdash A$	axiom H.	no goal created		By H we have the desired result.
	$H := A \to B$	$\vdash B$	elim H.	$H := A \to B$	$\vdash A$	By $A \to B$, it is enough to show A.
	$H := A \wedge B$	$\vdash B$	elim H.	no goal created		By H we have the desired result.
	$H := A \wedge B$	$\vdash A$	elim H.	no goal created		By H we have the desired result.
	$H := A \vee B$	$\vdash C$	elim H.	(1) $H := A$	$\vdash C$	Assume first A and show C.
			(or left H.)	(2) $H := B$	$\vdash C$	Assume next B and show C.
	$H := \exists x\, A[x]$	$\vdash C$	elim H.	$H := A[x]$	$\vdash C$	Let x be such that $A[x]$.
			(or left H.)			
	$H := \forall x\, A[x]$	$\vdash C$	apply H with t.	$H := \forall x\, A[x]$,	$\vdash C$	By H we have $A[t]$.
				$H_0 := A[t]$		
	$H := \neg A$	$\vdash C$	elim H.	$H := \neg A$	$\vdash A$	Since we have $\neg A$, it is enough to show A.
Others	$H := A \wedge B$	$\vdash C$	left H.	$H := A$, $H_0 := B$	$\vdash C$	By H we have A and we have B.
	$H := A \to B$,		apply H with H_0.	$H := A \to B$,		Since we have $A \to B$ and A we have B.
	$H_0 := A$	$\vdash C$		$H_0 := A$, $H_1 := B$	$\vdash C$	
		$\vdash A$	by_absurd.	$H := \neg A$	$\vdash A$	Do it by absurd and assume not A.
		$\vdash A$	elim False.		$\vdash$ *False*	Let us look for a contradiction.
	$H := \neg A$, $H_0 := A$	$\vdash C$	elim H with H_0.	no goal created		H and H_0 give a contradiction.
		$\vdash A$	Prove B.	(1)	$\vdash B$	Let us first show B then A (assuming B).
				(2) $H := B$	$\vdash A$	

3.3. The theorems proved

Then the students had three projects. Each of them took about ten hours. They had to do the proof with the computer and to write it, in the usual way, on paper. This work was evaluated and the corresponding mark was considered as the continuous assessment. The exam has been done in the same way: a proof to be done with the computer and, in the usual way, on paper. The files (questions and answers) of Heine, Noether and the exam can be found at the URL: http://www.lama.univ-savoie.fr/~david/PhoX.

Heine The goal was to prove Heine theorem: a continuous function on $I = [a, b]$ is uniformly continuous. It is not necessary to have "built" $\mathbb{R}$. It is enough to have, as lemmas, the properties of $\mathbb{R}$ that we need. It is in fact surprising that we actually need very few things: some properties of the order (totality, density, ...), its relation with the various operations ($\varepsilon > 0 \to \varepsilon/2 > 0, \ldots$) and, of course, the axiom of the least upper bound. The proof has been decomposed into two results:

– They first had to prove the compactness of the interval I in the following way: for every positive function lg there is a finite subset A of I such that each point of I is at a distance less than $\lg(x)$ from a point x in A. This proof was prepared by the introduction of the set P of points c in I for which there is a finite subset A of I such that each point of $[a, c]$ is at a distance less than $\lg(x)$ from a point x in A. The students had to prove, successively: (1) P is nonempty and bounded, (2) if m is its least upper bound, then $m \in P$ and finally (3) $m = b$.

– To prove Heine theorem the help was: take $\varepsilon > 0$, use the compactness with the function $\lg(x) = \alpha(x)/2$ where $\alpha(x)$ is the number given by the continuity of f at x with $\varepsilon/2$. This gives A. The number α we are looking for is $\min\{\alpha(x)/x \in A\}$.

The definition of continuity was given in a slightly unusual way in order to avoid the use of the axiom of choice which, with the usual definition, must be used and introduce a technical but uninteresting difficulty.

Noether The goal was to prove that, in a commutative ring with unit, the following properties are equivalent: (1) every increasing sequence of ideals is stationary, (2) every ideal is finitely generated.

The formalization does not raise any particular problem. In preparation of the main results, the students had to prove some useful lemmas, for example: if (I_n) is an increasing sequence of ideals then $\bigcup I_n$ is an ideal or if I is an ideal and $X \subset I$, then the ideal generated by X is a subset of I, etc. For the direction $(1) \Rightarrow (2)$, the proof is by contradiction: one has to find, by induction, an increasing sequence of finite sets. Moreover, this construction uses the dependent axiom of choice (a weak form of the axiom of choice). This needs a more expert knowledge of PhoX and the axiom of choice. Thus, the students were given the definition of the sequence, including its construction using the dependent axiom of choice.

Other theorems For the third project, the students were asked to propose results, taken from their other courses, they had not well understood and that they would like to work for better understanding. After some discussion (because most of the subjects they proposed

were only details in a proof) this gave the following subjects. Unlike for the previous projects, they had to do everything. Actually it was not so difficult since the preliminary work was, in fact, the same as the one for Heine theorem.

– Proof of the completeness of $\mathbb{R}$ by using the axiom of the least upper bound. There were three steps: (1) a Cauchy sequence is bounded, (2) a bounded sequence has an accumulation point (3) a Cauchy sequence converges to its accumulation point.

– Definition of $\mathbb{R}$ by using the Dedekind cuts and proof of some of its properties: $\mathbb{Q}$ is dense in $\mathbb{R}$, the axiom of the least upper bound, etc.

– Definition of $\mathbb{R}$ by completion of $\mathbb{Q}$, i.e., as set of equivalence classes of Cauchy sequences of rational numbers and proof of the completeness of $\mathbb{R}$.

– If a sequence of continuous functions is uniformly Cauchy, then it converges and its limit is continuous.

The exam Two proofs were given.

– If the union of two sub-groups is a sub-group, then one is included in the other.
– If f is continuous on $I = [a, b]$, then it is bounded on I.

In both cases, the proofs were "prepared". For the second we had given the intermediate results to be proved: let $P = \{c \in [a, b]/f$ is bounded on $[a, c]\}$. Show that P is upper bounded and nonempty and that its *lub* is in P. To be sure it will not be too long (they had three hours) and since the next step of the proof (the least upper bound is in P) was very similar, they were not asked to do it.

3.4. The results

The proofs with PhoX (1) All the students have completely done the proofs of the two first projects (Heine and Noether) with the computer. Some of them with very little help, some with more.

(2) For the last one, only the group with the fourth project finished. It was clearly the easiest one. After they finished, they were asked to prove that the set of piecewise linear functions on $[a, b]$ is dense in the set of continuous functions but they preferred to skip the class!

The group with the definition of $\mathbb{R}$ by Cauchy sequences did a good job but did not succeed in proving the completeness of $\mathbb{R}$, partly because of time. It was the best group but the project was probably too difficult.

The two other groups did rather few things, mainly because of the mathematical difficulties in their comprehension of what should be done.

(3) The results of the exam have been very disappointing. After 15 minutes, we had to tell them that, for the exercise on the groups, they have to use a proof by contradiction. There was actually no real choice but none of them had tried that. Only half of them finished the proof. The other half were blocked after having taken x_1 in $G_1 - G_2$ and x_2 in $G_2 - G_1$. For the exercise of analysis, except the best student who almost finished the proof, nobody succeeded in proving that the least upper bound of P was in P even though we told them many times that the proof was exactly the same as for Heine theorem. Some

of them, to prove $a \in P$, did not succeed in proving that f is bounded on $[a, a]$: they tried to use the continuity of f.

The proofs on paper After they had done the proof with PhoX, they were asked to write it on paper as they had done for a traditional homework. During the practical we helped them and answered their questions. Then, we corrected their first version (it was most often catastrophic) and explained, in detail, what was wrong. Finally, they had to give us a new version which was graded.

The results, for the proof of Heine theorem, were very disheartening: our impression was that the proof with the computer has been useless. There even has been a group that had written a proof of a result that looked like but did not correspond to the compactness of $[a, b]$. To help them, we took our proof of Heine and put, to the side of each command (or group of commands) a corresponding text in French. This helped them to understand that the formal proof is very close to the informal one.

Thankfully, this discouragement disappeared and, between the first and the last project, the improvement was clear: there was almost nothing to criticize on their last proof on paper.

3.5. The difficulties of the students

3.5.1. The difficulties to get a proof

To get a proof it is necessary (and sufficient) to use the rules. But,

The order in which the rules are applied is important For example, to prove $\exists x(A(x) \wedge B(x)) \to (\exists x A(x) \wedge \exists x B(x))$ the following list of commands `intro. intro. intro. left H. instance ?1 x.` is not the beginning of a correct proof (the hypothesis $\exists x(A(x) \wedge B(x))$ coming from the first `intro` has been denoted by H) and PhoX does not accept the last command. Indeed, this commands yield to the following sequence:

```
>PhoX> intro.
```
$\mathrm{H} := \exists x(A(x) \wedge B(x)) \vdash \exists x\, A(x) \wedge \exists x\, B(x)$
```
>PhoX> intro.
```
$\mathrm{H} := \exists x(A(x) \wedge B(x)) \vdash \exists x\, A(x)$
$\mathrm{H} := \exists x(A(x) \wedge B(x)) \vdash \exists x\, B(x)$
```
>PhoX> intro.
```
$\mathrm{H} := \exists x(A(x) \wedge B(x)) \vdash A(?1)$
$\mathrm{H} := \exists x(A(x) \wedge B(x)) \vdash \exists x\, B(x)$
```
>PhoX> left H.
```
$\mathrm{H} := A(x) \wedge B(x) \vdash A(?1)$
$\mathrm{H} := \exists x(A(x) \wedge B(x)) \vdash \exists x\, B(x)$
```
>PhoX> instance ?1 x.
Error: Fail match
```

It fails because the system keeps a constraint saying that `?1` should not use the variable x introduced by the command `left H` because the third `intro` introducing `?1` came before.

The correct list is: `intro. left H. intro. intro. instance ?1 x.` which yields to the same first goal but without the constraint.

This mistake, that will even not be detected in a proof on paper, is in fact crucial since it is similar to the very common mistake which consists, to prove that the sequence (u_n) is bounded, in taking a bound which depends on n.

Similarly, PhoX rejects: `intro. intro. intro. intro. instance ? x.` and answers that it does not know x. This remark of the machine is better accepted and understood by the students than if the teacher says something as: what is this x?

The choice of the rules that are applied is also crucial Because of the fact that proofs are very formal and it is always possible to give a command and see what the machine answers, the students search for proofs blindly. One could consider that this is a drawback of proof on computers, but this makes them realize that they have to stand back from what they are doing.

The following example (it is a real one) is a caricature but revealing. In the proof of Heine, the fact that $<$ is transitive on $\mathbb{R}$ was an axiom (claim or2 in the file Heine of Appendix B). To help the students to become more familiar with the software, they had to prove that $\leqslant$ is also transitive: $x \leqslant y$ is defined by (cf. line 4 of the file Heine) $x < y \vee x = y$. In fact, by the command `trivial`, PhoX does the proof itself. Some students were trying to prove $x = z$ from the hypotheses $x < y$ and $y < z$. This came from the fact that, in a proof of a $\vee$, the default answer to the command `intro` is the proof of the right-hand side of the $\vee$. Of course, they did not succeed but they had not understood, by themselves, that they had no chance to succeed.

This let us realize that there is a point on which it is necessary to insist: some rules are invertible (the new goal is provable iff the previous one is) but some others are not and thus, using them can give a dead end. For example, $\wedge_i$ is invertible but $\vee_i$ is not: to prove $A \vee B$ you must take the good choice! We also told the students that the absurdity rule can, in fact, be limited to atomic formulas (inequality, membership, etc.) or formulas starting with $\exists$, $\vee$.

Another example shows well this lack of distance. During the exam, on the exercise on the groups, a student had, in his hypotheses, $x.x^{-1} = x$. This was surprising and we thus asked him how he proved that. He told us that it was one of the given axioms. This let us realize that, effectively, one of the axioms for the inverse was written as $\forall x\, x.x^{-1} = x$. Since our proof had not used it, we had not seen the mistake but he had used it without scruples.

How to use the rules? The following heuristic works in most of the cases the students have to work on (both on the machine and on paper): first do all the (invertible!) `intro` then use a hypothesis by an `elim` or an `apply` and start again. The real difficulty should be to find the tricks that are necessary to get the proof. Most of the time, they have the good intuitions but it is the basic principles, i.e., what is a proof, that are not clear for them.

We had to repeat many times the role of the three basic commands: `intro` is used to "translate" the conclusion and does not care for the hypotheses. `elim H` uses H and gives a new conclusion (from which the previous one follows). `apply H` gives a new knowledge without changing the conclusion.

We had to repeat many times that a hypothesis of the form $\forall x(A[x] \to B[x])$ can be used only in the case an object t for which it is possible to prove $A[t]$ is available. For example, in the proof that two one-to-one functions with disjoint supports commute, some students were trying to use the injectivity of f at a point where the only thing known was $f(x) = x$.

We also had to repeat many times that to prove $\exists x A[x]$, the command `intro` can be used only when we know what is the object we are looking for. We also had to say that if one gives such an object by chance, it very rarely works! The following example, again very caricatured, is real: after having shown that a Cauchy sequence is bounded the students (in the proof of the completeness of $\mathbb{R}$) had to show the existence of an accumulation point. Despite the fact we had given them the idea to get this point, they had immediately given the command `intro` and they were pretending that the accumulation point they were looking for was the upper bound they had previously found.

The semantics of formulas In the proof that two one-to-one functions with disjoint supports commute, we have $H := \forall x(f(x) = x \vee g(x) = x)$ and we must show $\forall x(f(g(x)) = g(f(x)))$. We take an x by `intro`, we use H by `apply H with x` and consider the two cases by `elim H`. In the first case we thus have, for example, $H_0 := f(x) = x$. The students do not see clearly the difference between the use of H_0 with $g(x)$ (to get $f(g(x)) = g(x)$ which, of course, cannot be done) and the use of H with $g(x)$ which can be done but gives again two sub-cases.

Hypothesis versus conclusion The students discovered that, in a proof, the set of hypotheses may change and they must know, at each step of the proof, what can and what cannot be used. They sometimes confuse hypothesis and conclusion: we often found students trying to prove formulas which, in fact, are hypotheses!

The name given to objects Naming variables is another very important point. PhoX takes care of this and gives new names to the variables when it is necessary, i.e., for the rules $\forall_i$ and $\exists_e$. This eliminates an important source of errors: for example, it is impossible to do with PhoX a such a wrong proof of $\exists x A \wedge \exists x B \to \exists x(A \wedge B)$ since when we use the command `elim` first with $\exists x A$ and then with $\exists x B$, PhoX will give distinct names to the objects. Some years ago, the students were asked to do a formal proof of this result (on paper) and about half of them did succeed!

One may decide, at some point of the proof, to rename x in y if we know that y will no more be used. This is reasonable, and often done but, in the present version, there is no warning and we had the case of students who found that renaming y in x allowed to use the hypothesis $x = 0$ for y! They did not understand why the computer refused to prove $x = 0$ even if it was a hypothesis: the machine had not forgotten that there were two distinct x.

When the rules $\forall_e$ and $\exists_i$ are used, the machine asks for the objects by introducing a "?" and the user has to say what the "?" is. This helps to understand that, to use a hypothesis

as $\forall x\, A$ we must say which x is used and, similarly, when we want to prove $\exists x\, A$, we must say what is the x we are looking for and we can give it only when we, effectively, have found it.

Advantages and difficulties due to the software We try to give examples that can be treated without being an "expert" of the software. The first author is not such. The second author, PhoX's father, often does shorter or more elegant proofs than the former.

However, it happens that students are blocked because of the software. This is sometimes (but more and more rarely) because of a "bug" in PhoX, but this comes more commonly from the fact that they have to know a bit more about PhoX. The most significant example has been the use of the axiom of choice (AC) in the Noether example. Most of the time, in a mathematical proof, we do not mention the use of AC since its "truth" seems very natural. This is often a good thing because this difficulty is a secondary one. On the computer, this difficulty cannot be avoided because, if AC is necessary (in the mathematical sense of this word) its use must appear somewhere and it is not always easy. The chosen presentation was too subtle but, now, we have understood the way it should be done: we should provide the form of AC they really need. For instance, $\forall x \in A \exists y \in B\, P(x, y) \to \exists f \forall x \in A(f(x) \in B \wedge P(x, f(x)))$ for AC and $a \in A \wedge \forall x \in A \exists y \in A\, P(x, y) \to \exists u(u(0) = a \wedge \forall n \in \mathbb{N}(u(n) \in A \wedge P(u(n), u(n+1))))$ for the axiom of dependent choice. These results (which are not trivially deduced from the form of AC presently given with PhoX) will be included in the library of the next version.

On the machine everything must be very precise: this has advantages and drawbacks.

– Advantages: requiring the students to write definitions by themselves is very formative and it took them often a long time. The following fact (that happens quite often) also is very revealing. Students are blocked on the proof of some point. They complain on the machine: "Why PhoX does not accept? It is trivial". It is indeed easy but, nevertheless, we ask them to be a bit more precise: most often, the arguments they give are totally wrong!

The fact that the machine has a typing algorithm, like in programming languages, is very useful and forbids meaningless formulas. The students who worked on the definition of $\mathbb{R}$ either by cuts or by Cauchy sequences took benefit of that because they had to use various types of objects.

– Drawbacks: anything that has been forgotten, even if it is not important, obliges to restart at the beginning and, if you are not careful enough, it is often with some difficulties: for example, in a definition on sequences, the lack of $n \in \mathbb{N}$ that is usually implicitly required a great deal of time (with PhoX this is not handled by the type system, because we have partial functions).

3.5.2. Difficulties to write a proof

In a proof on paper, the reader must be able to understand the succession of the given arguments. In this case, it is not very difficult to indicate the errors, to show that an assertion is not correct by giving a counter-example, to adjust something imprecise, etc. Otherwise, and it is often the case in bad papers, the reader is disarmed and can only say that he does not understand anything and thus cannot help the student.

As already mentioned, the first papers were very bad. But, the comparison between the proof done on the machine (that is necessarily correct since PhoX cannot do a wrong

proof) and its translation on paper allowed to tell them, if what they had written was not satisfying: "what you have written is not what you did on the machine". This permitted, gradually, to improve their proofs on paper.

The main remaining difficulties are the following.

The good level of detail In many papers, the proof of $\{a\} \subset [a, b]$ took about ten lines: that came from the fact that, on the machine, the students had used about ten commands even though a `trivial` was enough (both on paper and on computer). Another example: on the machine, when they had to consider differently the two cases $x < m$ and $m \leqslant x$, they had to use the axiom saying that the order on $\mathbb{R}$ was total but, on paper, it was enough to say: 1st case: $x < m$, 2nd case: $m \leqslant x$. However the reader will note that, in a similar proof on an ordered set, a paper that had not explicitly mentioned the fact that these two cases can be considered because the order is total would probably be criticized.

Being able to see what does not need to be detailed (because it is considered as easy) and what must be is, intrinsically, not simple and also depends on the teacher: there is no rule! But this is probably not a major problem because the answer comes with the experience and it is easier to learn how to give less details than to give more.

The semantics versus the syntax of formulas The machine does not confuse $\forall x(A[x] \to B)$ with $\forall x A[x] \to B$: in response to the command `intro` it asks to prove $A[x] \to B$ in the first case and B (with hypothesis $\forall x A[x]$) in the second case. This distinction is not so clear for the students. In the Noether example (every ideal is finitely generated iff every increasing sequence of ideals is stationary) the proof on paper of several students let us think they had made a similar mistake. We asked them to be more precise. Despite the fact they had done the correct proof on the machine, their answers clearly showed they had made the confusion!

Mixing mathematical formulas and natural language In sentences as (this example is taken from the construction of $\mathbb{R}$ by the cuts) "I know that $x < y$ and $y < z$ thus $x < z \Rightarrow x \in f(z)$" are difficult to decode and the students can have the same kind of confusion as above. It is thus also necessary that students learn how to go, precisely, from the natural to the symbolic language (and vice versa). But again, this is probably a question of experience and thus a minor problem.

3.6. The point of view of the students

Below is the questionnaire given for the evaluation of this course. For each question, the mean value of the answers is given. This value is, of course, not very meaningful from a statistical point of view since only eight students have chosen this course. However, since the answers are rather homogeneous, there are good reasons to be optimistic.

The questionnaire The goal of this course was to teach you the way to do correct proofs in mathematics. Please, answer to the following questions. This will help us to improve the course for next years.

1) This course has not been a course of logic but only a help to learn the way of reasoning. Would you have wished also a more classical part introducing the basis of mathematical logic (as, for example, in [3]): 7 yes and 1 no.

2) In the following questions, give a score between 0 and 5: 0 = not at all, 5 = yes very much. Do you think that:

- this course has helped you in the comprehension of what a proof is: 3.9;
- this course has helped you to write proofs on paper: 3.4;
- the fact of doing very detailed proofs with the machine has been important in this learning: 3.6;
- learning the commands of PhoX is difficult: 2.0;
- to use PhoX is difficult: 2.2;
- this course should be mandatory: 3.1;
- we should use PhoX with the first year students on simpler examples: 3.2;
- you will recommend this course to the students if it is still optional next year: 3.8.

4. The main points of the other years

4.1. Logic and/or PhoX

When we started to use PhoX with the students, this was simply to be able to write proofs a bit too long to be written on paper but these proofs were toy examples, i.e., the kind of proofs usually given in a logic course. The main goal of the course was an introduction to logic. As we already said, we gradually thought that doing real proofs (the kind of proofs students find in a course of algebra or topology) in detail may help them to understand what a proof really is and also to help them to correct the numerous mistakes of reasoning they do. We thus started to give them real proofs.

At that time, we started with the syntax of first order logic and we introduced the rules of natural deduction. It seems (see Section 4.3) that this is not necessary if the goal is to make proofs. However is it reasonable to teach mathematics without defining first, even informally, what is a proposition and a proof, since they are the basis of all the mathematics? Nevertheless, this is done (at least in France) most of the time. When we do not define these concepts, we enforce a division between the students who discover their meaning by themselves (this is what happened to those who became teachers!) and the others. Here is a "real" example: a second year student, who just learned that all symmetric matrices can be diagonalized, did not consider for certain that he would never have the "bad luck" to see a counter-example.

4.2. The choice of examples

The main difficulties, for us, are to find examples for which:

- The formalization is very close to the informal reasoning that is given in the course of analysis or algebra.

– The properties of the objects that are considered do not need a complicated axiomatization.
– The reasoning, i.e., the chain of $\forall$ and $\exists$ is, from an educational point of view, interesting.

Note that it is easier to find examples from analysis than from algebra and that these results are often educationally more interesting because the alternation of three quantifiers($\forall\varepsilon\exists\alpha\forall x\ldots$) in formulas is one of the main difficulties for the students. Until now we never gave examples from linear algebra. This simply comes from the fact that the manipulation (which is considered as trivial by everybody) of finite sums of reals or vectors is not so easy when one tries to formalize it. Since this is a field with a lot of nontrivial exercises, it will be interesting to find a good solution.

The first "real" example we have done with PhoX was a proof of the intermediate value theorem. Even though we had decomposed the proof into a list of rather simple lemmas we realized (but too late) that is was too difficult for them: after more than an hour none of them had been able to prove the first lemma (if f is continuous and $f(x) > c$, there is an $\alpha > 0$ such that $f(y) > c$ for all y in $]x - \alpha, x + \alpha[$). This is probably not because of PhoX: they would also had been unable to write a precise proof on paper.

Here are the most significant examples the students have done:

– the uniqueness of the limit,
– the closure of a union equals the union of the closures (in a metric space),
– the definition of continuity with strict inequalities (for example $|x - y| < \alpha$) is equivalent to the definition with nonstrict inequalities (i.e., $|x - y| \leqslant \alpha$),
– the image of a connected set by a continuous function is connected (in a metric space),
– two permutations of a set E with disjoint supports commute,
– if the union of two subsets of $\mathbb{R}$ is unbounded, then one of them is unbounded,
– in a ring a prime ideal is irreducible,
– in a principal ring, an ideal is maximal iff it is prime.

4.3. Other experiments

First year students During the last two years we had done simpler examples with students who were in first year at the university. They essentially had to prove equalities between sets: for example $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ or $f(A \cap B) \subset f(A) \cap f(B)$. The most difficult example was the proof of the uniqueness of the limit. We then realized that students that had been able to do the proof with PhoX were unable to write it correctly "in French".

This experiment has been too short and with too few students to be able to give some conclusions but the students (some of them were good in math but some others were among the weakest) told us they had learned much.

However this has been the starting point for the experiment detailed above and we decided to do more with the students in third year and to replace completely the logic course by a training in mathematical reasoning: they were asked to do proofs on the machine and then to "translate this proof" in French, on the paper.

The last experiment The course given in 2002–2003 is now suppressed: officially not because it is logic but because mathematical reasoning is not considered as a priority! In the short course (only 10 hours) that has been maintained in 2003–2004, we had not enough time to give the same introduction as before. The learning of PhoX has been done in the following way: we gave the students the table of the main commands (see Table 1) and the file below (and two other similar ones) and told them: try to understand how it works. You then will have to do the same thing by yourself. We were happy (and even rather surprised) that it worked perfectly well since, from the questionnaire we gave them, it appears that they have found that PhoX was easy to learn and to use. They succeeded quite well with the three mathematical proofs we gave them. Maybe these proofs were not difficult enough. It was: (1) the equivalence of continuity by inverse image of open sets and the usual definition with ε, (2) the equivalence of two definitions of the closure of a set and (3) in a ring a prime ideal is irreducible. But the tactic `trivial` was working too much and some students asked us, after `trivial` had worked ... what was the proof! They also had less time to write the proofs "in French" and this remains a major difficulty for them: what they write is often a paraphrase of the commands. Doing a synthesis, understanding what has to be said and what can be ignored needs a longer time.

A file to start with PhoX The comments in italic are the explanations to understand the proof. The students are asked to do the same thing in their proofs.

Sort E.
def Infix[3] A "union" B $=\lambda$x (A x or B x).
def Infix[3] A "inter" B $=\lambda$x (A x and B x).
def Infix[5] A "subset" B $=\forall$x: A B x.
Cst open: (E $\rightarrow$ prop) $\rightarrow$ prop. (*note that "open" is a symbol and is not defined*: *this is simply because its definition is useless for our goal*!)
def inverse f A x = A (f x).
def image f A x $=\exists$y: A x = f y.
def connected A $=\forall$U,V: open (A subset (U union V) $\rightarrow$ $\forall$x ((U inter V) x) $\rightarrow$
(A subset U or A subset V)).
def continuous f $=\forall$A (open A $\rightarrow$ open (inverse f A)).
goal $\forall$f,A (continuous f $\rightarrow$ connected A $\rightarrow$ connected (image f A)).
intro 4. (*let f be continuous and A be connected*)
intro 6. (*let U and V be open sets with empty intersection. Assume* $f(A)$ *is included in U union V. Let us show that* $f(A)$ *is included in U or in V*)
apply H with H1. apply H with H2. (*since f is continuous and U, V are open,* $f^{-1}(U)$ *and* $f^{-1}(V)$ *are open*)
local U1 = inverse f U. local V1 = inverse f V.
prove A subset U1 union V1. (*let U1* $= f^{-1}(U)$ *and V1* $= f^{-1}(V)$; *let us show that A is included in U1 union V1*)
trivial = H3. (*this comes immediately from the fact that* $f(A)$ *is included in U union V*)
prove $\forall$x (U1 inter V1) x. (*let us show that U1 inter V is empty*)
trivial = H4. (*this comes immediately from the fact that U inter V is empty*)
prove A subset U1 or A subset V1. (*let us show that A is included in U1 or in V1*)

```
apply H0 with G1 and G2.
trivial = H H1.
trivial = H H2.
axiom G3. ( this comes from the connectivity of A used with U1 and V1)
trivial = G3. ( we thus get the desired result)
save thm.
```

5. Conclusion and perspectives

Even if the results are not spectacular, this experiment seems to be extremely positive: it helped us to better understand where the major difficulties of the students are and it made them progress. For us, it has been very interesting and stimulating. The course required more work to teach (much more than for a classical course of logic) but the reward was to have done something that made the students progress.

The simple fact of being able to tell the students precisely what is a proof is already very useful. As long as one manipulates only simple sentences, it is not necessary to know the underlying grammar but, when we want to understand sentences with a more complicated structure, the words to analyze them become necessary. We think it is the same thing for proofs. A very detailed proof with a computer allows one to be more precise about the difficulties and this helps to solve them. At least the students have understood that, in a proof, after having done every invertible `intro`, it is necessary to think to find the hypothesis on which an `elim` can be done.

The association of the work with the machine and on paper allowed us to understand that finding a proof and writing it are two very distinct things: a student can understand a proof and be unable to write it and vice versa. We thus must teach the way of writing proofs. Usually the teacher spends very little time to help the students to write proofs, to correct in detail what they have done, to suggest improvements, etc. The possibility to compare what is written on paper with the proof they have done with the machine is very helpful for that.

Only a few students took our (optional) courses because they are in competition with courses on geometry and probabilities that both were dedicated to prepare the Capes (the examination to become a teacher in secondary schools) and the students who intend to prepare this examination considered the latter to be more useful. However, we believe it has been very profitable for them. We are convinced that, if students who did not attend our courses were asked to do the proofs we have been working on, the results would be catastrophic. Will these courses be beneficial for the other courses? Will they keep, later on, the good manners they got? The colleague who has given the course of algebra and did not know who, among his students, was attending our course, told us several times that he could guess just by seeing the way they did their proofs in his course. We must confess that he also is logician and that his opinion can be sympathetic towards logic.

It is nevertheless clear that some improvements are necessary. The interface of PhoX should be improved, it would also be nice if students could write proofs in French (or English) instead of writing commands. In this direction, we have a research project whose

goal is that PhoX "understands natural language", i.e., that, instead of using "intro" or "elim", one can use "let x be positive, show that ...".

If we had enough money (this is not the case today!) to give such a course during the two first years of undergraduate studies, should we do it for all the students or only for those who want to do mathematics? Being able to reason correctly is useful for everybody and the fact that it is somehow a game would probably be more important than the formal side (at least for the students who are not discouraged by computers) but does it have higher priority than the technique of computations with which they are not familiar enough?

The teachers in the universities complain about the fact that students coming from secondary schools cannot reason correctly. Is it useful to have such courses in these schools?

If we want to progress, it seems necessary that this experiment keeps going. Unfortunately, it is possible that it will stop there. The mathematicians in general and our colleagues in particular do not like logic and want to throw it away from the mathematical studies. The reader will appreciate the humor of the arguments they give: the learning of reasoning is not a part of the program of the Capes examination, this course comes too late. This is too bad because we really believe that doing formal proofs on computers will help the student to progress.

For further reading

The following references could also be of interest to the reader: [2,4–6].

Appendix A. Introduction to PhoX

PhoX is available (and free) on the web page of C. Raffalli. Its logic is higher order logic. The *typed* aspect allows to distinguish between type errors (for example, $\lambda + x$ where λ is a scalar and x a vector) and the errors of reasoning. There is no problem with the students since it is their "natural" language.

PhoX uses a functional notation, for example dxy for $d(x, y)$ or $Imfy$ for $y \in Im(f)$ or open $U = \forall x(Ux \to \exists e > 0\ \forall y(dxy < e \to Uy))$. This notation is not a problem for the students but, unfortunately, it is not the case for some of our colleagues (not logicians)! It would be very easy to define an infix symbol for the membership relation (this is the main missing symbol) to be able to write $y \in Im(f)$ instead of $Imfy$ but this is really unnecessary with students. The notations for the formulas do not introduce particular problems because they use traditional mathematical symbols (except conjunction and disjunction). PhoX provides useful abbreviations such as $\forall x, yA$ for $\forall x \forall y A$, $\forall x \in A B$ for $\forall x(Ax \to B)$ or $\exists x < yB$ for $\exists x(x < y \wedge B)$ as well as traditional priorities to avoid numerous parentheses. The common associativity rule for the arrow ($A \to B \to C$ is read as $A \to (B \to C)$) is very unfamiliar to the students and the fact that it is not equivalent to $(A \to B) \to C$ is not always clear for them.

The interface is made with XEmacs [1] and ProofGeneral [15] of D. Aspinall. Fig. A.1 shows an example of PhoX's screen. The screen is divided into two parts: the upper part

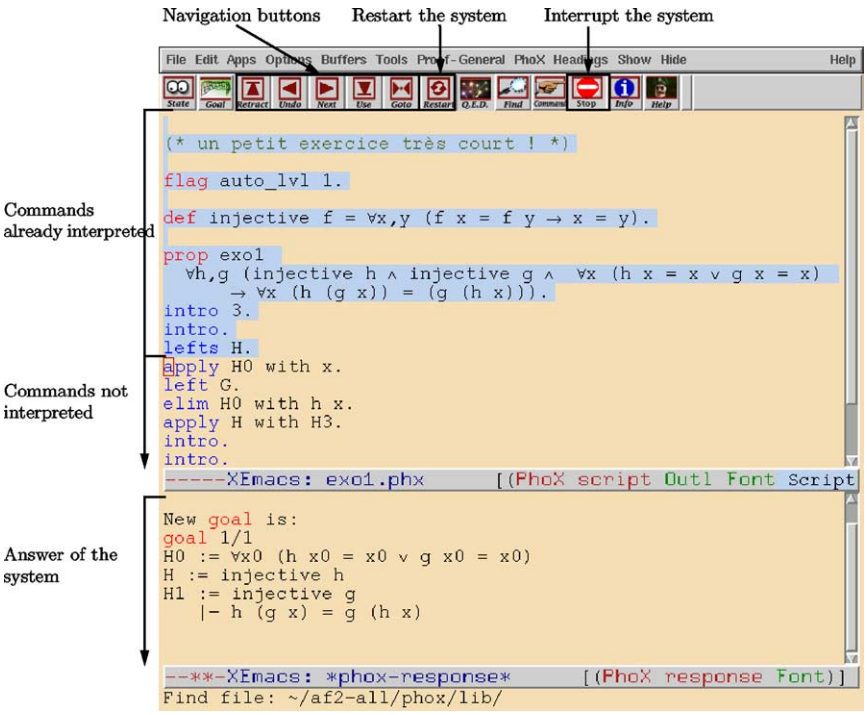

Fig. A.1. A typical PhoX screen.

contains the *script* of the commands given to the system and the lower part, PhoX's response to the last command.

Appendix B. An example of file

The file below is the one for the proof of Heine theorem. This is exactly the one (translated in English) on which the students have worked. We have only deleted the preliminary lemmas (mainly about the order) given to help the students to get more familiarity with PhoX.

The commands beginning with `Cst` allow to introduce new symbols, those beginning with `def` give definitions and those beginning with `claim` are axioms or, more precisely, the properties (here on the reals) that we need to do the proof. The commands beginning with `prop` introduce the results to be proved and those beginning with `prove` are intermediate lemmas.

Some formulas are written as $A \to B \to C$ instead of the (more common) $A \wedge B \to C$: this is simply because this makes the proofs (a bit) shorter in PhoX.

Heine

Sort real.

```
Cst Infix[5] x "<" y : real → real → prop.
def Infix[5] x ">" y = y < x.
def Infix[5] x "⩽" y = x < y ∨ x=y.
def Infix[5] x "⩾" y = y ⩽ x.
Cst lInfix[3] x "-" y : real → real → real.
Cst lInfix[3] x "+" y : real → real → real.
Cst lInfix[2] x "/" y : real → real → real.
Cst zero : real.
Cst one : real.
Cst two : real.
Cst a : real.
Cst b : real.
Cst abs : real → real.
def I x = a ⩽ x ∧ x ⩽ b.
def maj m X = ∀y ∈ X y ⩽ m.
def Maj X = ∃m maj m X.
def sup m X = maj m X ∧ ∀y (maj y X → m ⩽ y).
def empty = { x | False}.
def plus X a = { x | x ∈ X ∨ x = a }.
Inductive Finite X = Empty : Finite empty
| plus : ∀X,a (Finite X → Finite (plus X a)).
def Image A h = { y | ∃x ∈ A y = h x }.

claim or1 ∀x,y ¬ (x < y ∧ y < x).
claim or2 ∀x,y,z (x<y → y<z → x<z).
claim or3 ∀x,y (x ⩽ y ∨ x > y).
claim or4 ∀x,y (x<y → ∃z (x<z ∧ z < y)).
claim or5 ∀e>zero ∀x x > x-e.
claim or6 ∀e>zero ∀x x < x+e.
claim or7 ∀x x - x =zero.
claim or8 one > zero.
claim or9 ∀x,y,x0,y0 (x<x0 → y<y0 → x+y<x0+y0).
claim or10 a < b.
claim abs1 ∀x abs zero = zero.
claim abs2 ∀x,y abs (x-y)= abs (y-x).
claim abs3 ∀e>zero ∀x,m (x⩽m → x>m-e → abs(x-m) <e).
claim abs4 ∀e>zero ∀x,m (x<m+e → x>m → abs(x-m) <e).
claim abs5 ∀x,y,z abs (x-y) ⩽ abs (x-z)+ abs(z-y).
claim axlub ∀X (∃x X x → Maj X → ∃m (sup m X)).
claim div1 ∀x>zero x/two > zero.
claim div2 ∀x>zero x/two < x.
claim div3 ∀x x/two + x/two=x.

prop compact ∀h (∀x h x > zero → ∃A Finite (A ⊂I ∧ ∀x(a⩽x → x⩽b → ∃y ∈ A abs
(x-y) < (h y)))).
intro 2.
```

local P c = I c ∧ ∃A Finite (A ⊂ I ∧ ∀x(a ⩽ x → x ⩽ c → ∃y ∈ A abs (x-y) < (h y))).
prove P a. ...
prove maj? P. ...
prove ∃m sup m P. ...
left G1. prove I m. ...
prove P m. ...
prove ¬ m<b. ...
prove m=b. ...
... save.

def continuous f = ∃g (∀e>zero ∀x (g e x) >zero ∧ ∀x ∈ I ∀e > zero ∀y ∈ I (abs (x - y) < g e x → abs (f x - f y)< e)).
def UC f = ∀e>zero ∃c>zero ∀x,y ∈ I (abs (x-y) <c → abs (f x-f y) <e).

prop Heine ∀f (continuous f → UC f).
intro 2. left H. rename x g. left H.
intro 2. local e1 = e/two. local h x = (g e1 x) / two.
apply compact with h.
... save.

Appendix C. An example of proof

The example given below is typical of those which can be done with the students. It has been treated by them. The goal is to prove that two definitions of the continuity of a function are equivalent: this equivalence, obvious for the teacher, is not at all immediate for the students. We give only one of the directions, the other is similar. We have written it in a rather elaborate way in order to show the possibilities of the system. In practice, the students make longer proofs by breaking up some commands with more elementary ones. Note that the first part of the example is prepared by the teacher: the work of the student begins only after the first paragraph. The prompt `>PhoX>` appears before each command and the answer is given below.

```
>PhoX> Sort real.
>PhoX> Cst Infix[5] x "<=" y : real -> real -> prop.
>PhoX> Cst Infix[5] x "<" y : real -> real -> prop.
>PhoX> def Infix[5] x ">" y = y < x.
>PhoX> def Infix[5] x ">=" y = y <= x.
>PhoX> Cst d : real -> real -> real.
>PhoX> Cst 0 : real.
>PhoX> def continuous1 f x = ∀e>0 ∃a>0 ∀y(d x y < a → d(f x)(f y) < e).
>PhoX> def continuous2 f x = ∀e>0 ∃a>0 ∀y(d x y ⩽ a → d(f x)(f y) ⩽ e).
>PhoX> claim lemme1 ∀x, y(x < y → x ⩽ y).
>PhoX> claim lemme2 ∀x>0 ∃y>0 ∀z(z ⩽ y → z < x).
>PhoX> goal ∀x, f(continuous1 f x → continuous2 f x).
goal 1/1
    ⊢ ∀x, f(continuous1 f x → continuous2 f x)
```

```
>PhoX> intro 5.
goal 1/1
H := continuous1 f x
H0 := e > 0
    ⊢ ∃a>0 ∀y(d x y ⩽ a → d(f x)(f y) ⩽ e)
>PhoX> apply H with H0. rmh H H0.
goal 1/1
G := ∃a>0 ∀y(d x y < a → d(f x)(f y) < e)
   ⊢ ∃a>0 ∀y(d x y ⩽ a → d(f x)(f y) ⩽ e)
>PhoX> lefts G $∃ $∧.
goal 1/1
H := a > 0
H0 := ∀y(d x y < a → d(f x)(f y) < e)
    ⊢ ∃a0>0 ∀y(d x y ⩽ a0 → d(f x)(f y) ⩽ e)
>PhoX> apply lemme2 with H. rmh H.
goal 1/1
H0 := ∀y(d x y < a → d(f x)(f y) < e)
G := ∃y>0 ∀z⩽y z < a
   ⊢ ∃a0>0 ∀y(d x y ⩽ a0 → d(f x)(f y) ⩽ e)
>PhoX> lefts G $∃ $∧
>PhoX>rename y a'.
goal 1/1
H0 := ∀y(d x y < a → d(f x)(f y) < e)
H1 := a' > 0
H2 := ∀z⩽a' z < a
    ⊢ ∃a0>0 ∀y(d x y ⩽ a0 → d(f x)(f y) ⩽ e)
>PhoX> intros $∀ $∃ $∧ $→.
goal 1/2
H0 := ∀y(d x y < a → d(f x)(f y) < e)
H1 := a' > 0
H2 := ∃z⩽a' z < a
    ⊢ ?1 > 0
goal 2/2
H0 := ∀y(d x y < a → d(f x)(f y) < e)
H1 := a' > 0
H2 := ∀z⩽a' z < a
H3 := d x y ⩽ ?1
    ⊢ d(f x)(f y) ⩽ e
>PhoX> axiom H1. auto +lemme1.
```

Appendix D. Why PhoX?

The same course could probably be done using other systems but we believe PhoX is one of the easiest to learn especially if you do not know logic. For instance, if we compare with Coq (the most popular system in France), we can give the following arguments:

- Coq has many commands to do proofs (tactics) and even if most proofs can be carried out with a small set, this set will depend on the teacher! In PhoX we really use about twelve tactics with students including the tactics for equational reasoning. The total number of tactics in the current 0.83 version is 19. Moreover these tactics are extensible and contextual. For instance, the `left H` command can be described as "simplify the hypothesis `H`". If `H` is $A \wedge B$, PhoX will replace `H` by two hypotheses A and B but, if `H` is $x + 3 = y + 3$, it will transform `H` into $x = y$. The same command can be used in many situations and can be modified by the teacher using the `new_elim` command.
- In Coq there is a difference between intensional and extensional equality whereas, in math, we use only the second. Thus, equational reasoning is not so easy to do with Coq. PhoX uses higher-order unification modulo the known (conditional) equations and can perform proofs with very simple equational reasoning without using their specific commands.
- The calculus of construction (compared with HOL) is probably too complicated for math students.

Making PhoX easy to learn has always been the first priority in its development and a lot of improvements have been introduced since our first teaching experiment. To use a software that we maintain by ourself is thus a good thing.

References

[1] The XEmacs editor, www.xemacs.org.
[2] L. Damas, R. Milner, Principal type schemes for functional programs, in: Ninth Annual ACM Symposium on the Principles of Programming Languages, ACM, 1982, pp. 207–212.
[3] R. David, K. Nour, C. Raffalli, Introduction à la logique, Masson, 2001.
[4] Y. Delmas-Rigoutsos, R. Lalement, La Logique ou l'Art de raisonner, Le Pommier, 2000.
[5] G. Dowek, La Logique, Flammarion, 1995.
[6] R. David, C. Raffalli, Apprentissage du raisonnement assisté par ordinateur, Quadrature No. 45, printemps 2002.
[7] J. Harrison, HOL Light, www.cl.cam.ac.uk/Research/HVG/HOL.
[8] INRIA, The Coq Proof Assistant, coq.inria.fr.
[9] S. Konrad, et al., HOL98, www.cl.cam.ac.uk/Research/HVG/HOL.
[10] J. Strother Moore, et al., ACL2, www.cs.utexas.edu/users/moore/acl2.
[11] S. Owre, N. Shankar, PVS, pvs.csl.sri.com.
[12] L. Paulson, et al., Isabelle, www.cl.cam.ac.uk/Research/HVG/Isabelle.
[13] R. Pollack, LEGO, dcs.ed.ac.uk/home/lego.
[14] C. Raffalli, PhoX, www.lama.univ-savoie.fr/~raffalli/phox.html.
[15] K. Thomas, D. Aspinall, et. al., Proof General, zermelo.dcs.ed.ac.uk/~proofgen.